732A99 ComputerLab2 Block1

Namita Sharma

12/8/2019

Assignment 2 - Analysis of Credit Scoring

2.1 Import data and divide into train, validation and test datasets

The creditscoring.xls data in imported and divided into training, validation and test sets in the proportion (50%, 25%, 25%) using R code detailed in appendix 2.1.

2.2 Fit a decision tree to training data using different measures of impurity

2.2 (a) Deviance

Table 1: Confusion Matrix Training Data

	bad	good
bad	61	86
good	20	333

Table 2: Confusion Matrix Test Data

	bad	good
bad	28	48
good	19	155

Misclassification Rates

	Training	Test
Misclassification Rate	0.212	0.268

2.2 (b) Gini index

Table 3: Confusion Matrix Training Data

	bad	good
bad	66	81
good	38	315

Table 4: Confusion Matrix Test Data

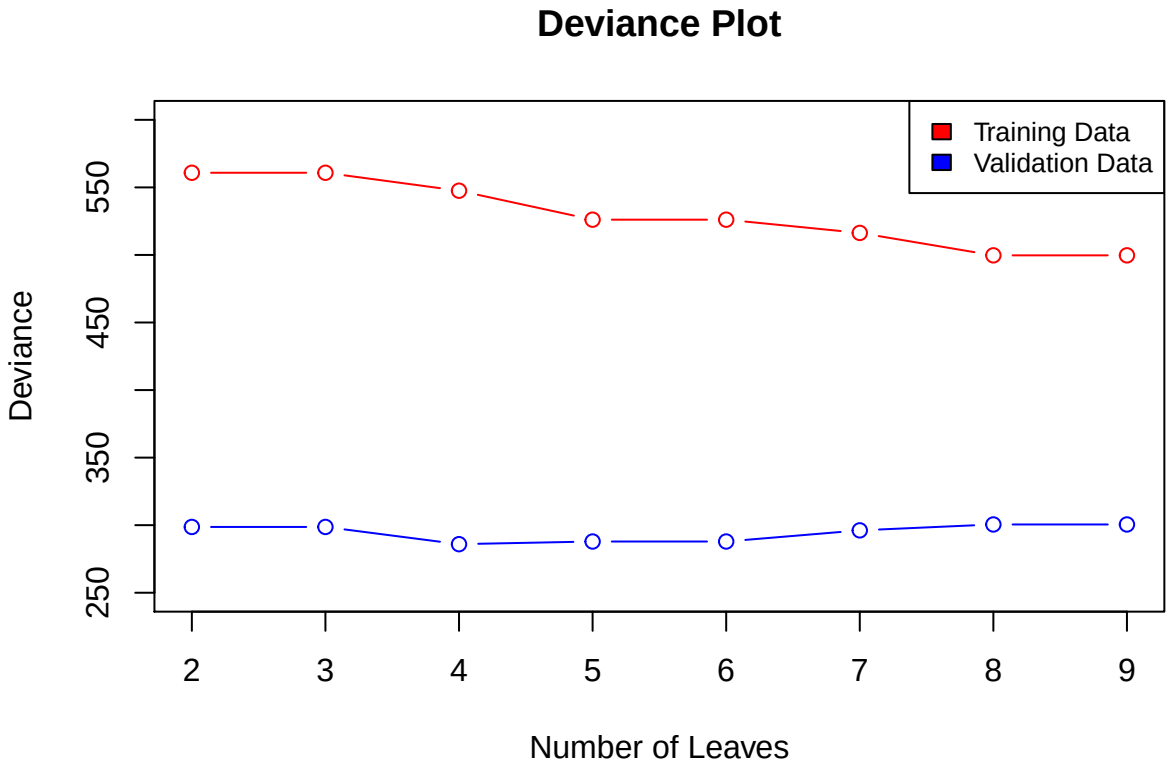
	bad	good
bad	18	58
good	35	139

Misclassification Rates

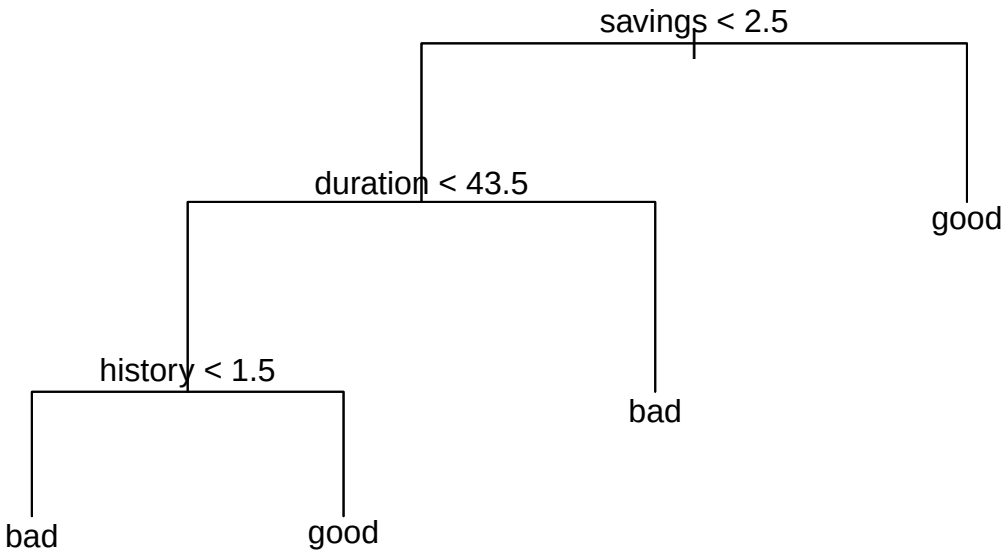
	Training	Test
Misclassification Rate	0.238	0.372

Fitting a decision tree using deviance as th impurity measure gives a better / lower misclassification rate.

2.3 Selecting optimal tree by train and validation




```
## Variables used in tree construction :  savings, duration, history
## Optimal tree depth :  3
```



It can be seen from the graph that the optimal tree has 4 terminal nodes and uses three of the variables for splitting the nodes. The 3 variables used are savings, duration and history. The misclassifcation rate of the optimal tree on the test data is 0.256.

2.4 Classification using Naive Bayes

Table 5: Confusion Matrix Training Data

	bad	good
bad	95	52
good	98	255

Table 6: Confusion Matrix Test Data

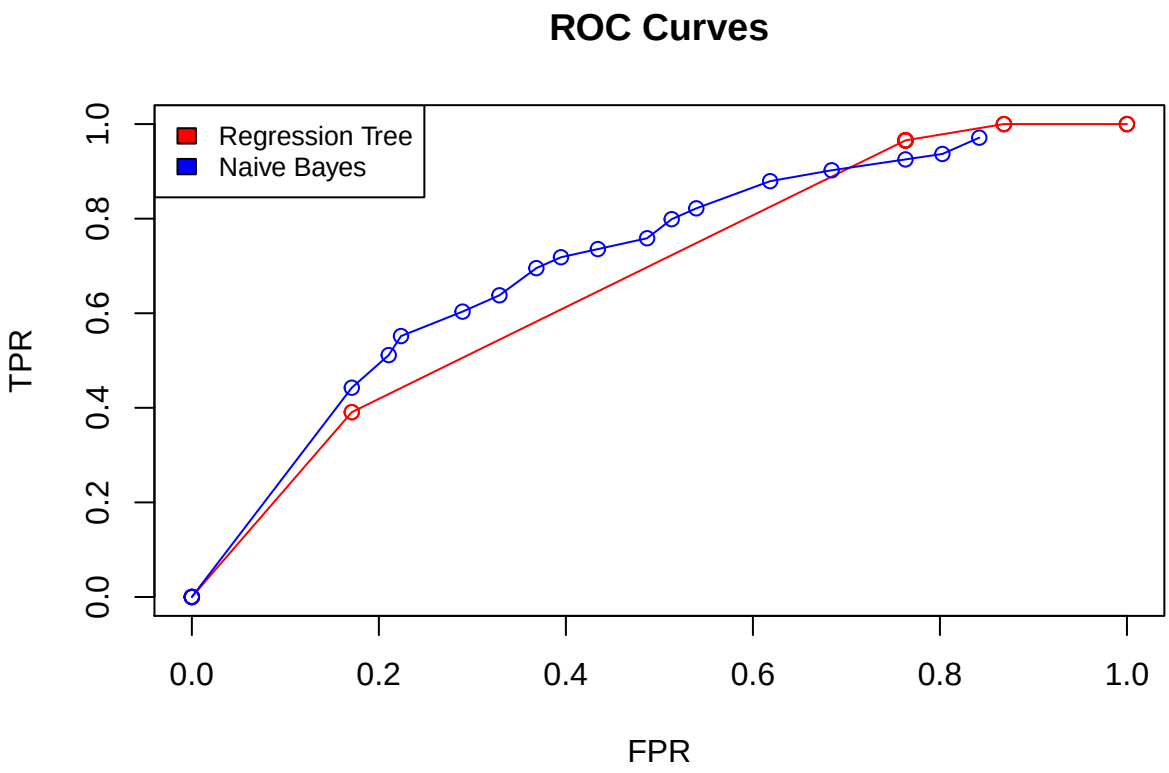
	bad	good
bad	46	30
good	49	125

Misclassification Rates

	Training	Test
Misclassification Rate	0.3	0.316

The misclassification rate on the test data is higher in case of naives bayes classification when compared to the optimal tree classification.

2.5 Compare Optimal Tree and Naivebayes models and plot ROC



The red ROC curve represents the TPR and FPR for the optimal regression tree classifier and the blue ROC curve gives the TPR and FPR for the naive bayes classifier. It can be seen that the area under the curve (AUC) is greater for the blue curve than the red and hence, for the same FPR, the former gives a higher TPR than the latter. Depending on how many False Positives are acceptable, it may be said that the naive bayes is a better classifier than the optimal regression tree.

2.6 Naive Bayes classification with a loss matrix

Table 7: Confusion Matrix Naive Bayes Classification without loss function

	bad	good
bad	46	30
good	49	125

Table 8: Confusion Matrix Naive Bayes Classification using loss function

	bad	good
bad	71	5
good	122	52

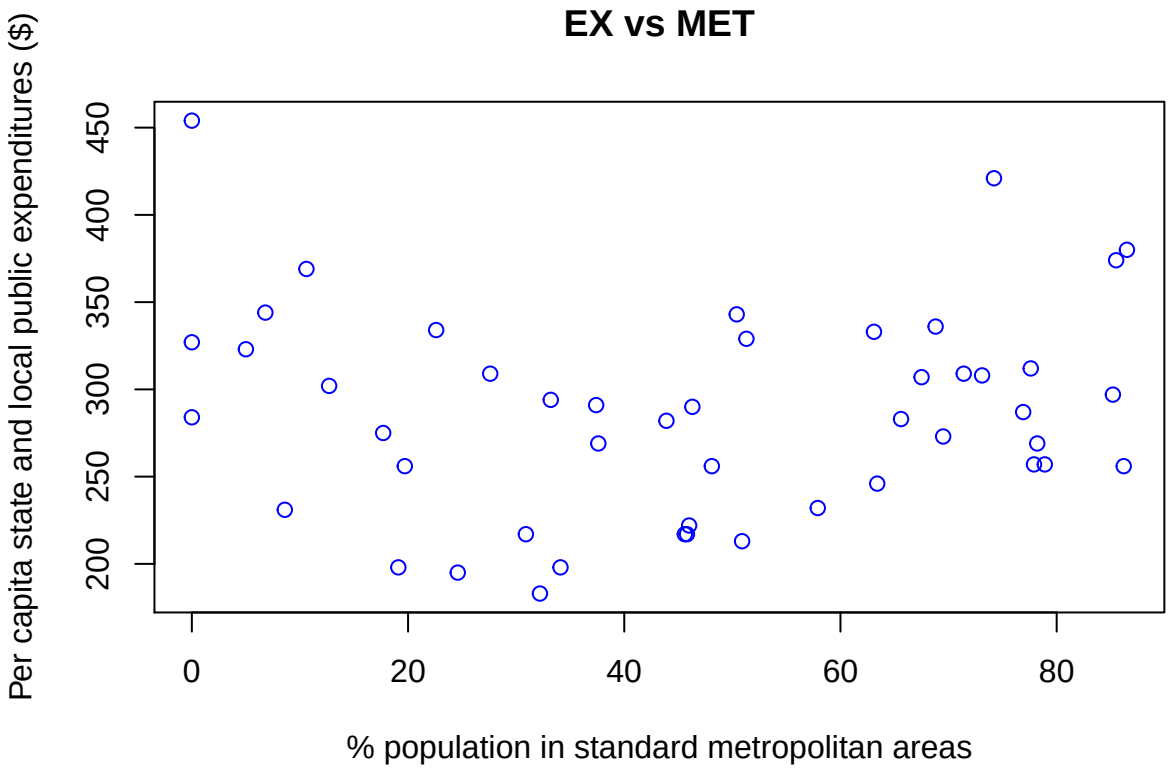
Misclassification and False-positive Rates Comparison

	NaiveBayes	NaiveBayesLoss
Misclassification Rate	0.316	0.508
False-Positive Rate	0.395	0.066

The misclassification rate is higher when the naive bayes classification uses the loss matrix. However, the false positive rate is much smaller when compared to naive bayes classification with even loss. Because the penalty for classifying a bad customer as good is very high, the model uses a higher threshold for classifying a customer as "good". As a result, we see an decrease in the true and false positive rates.

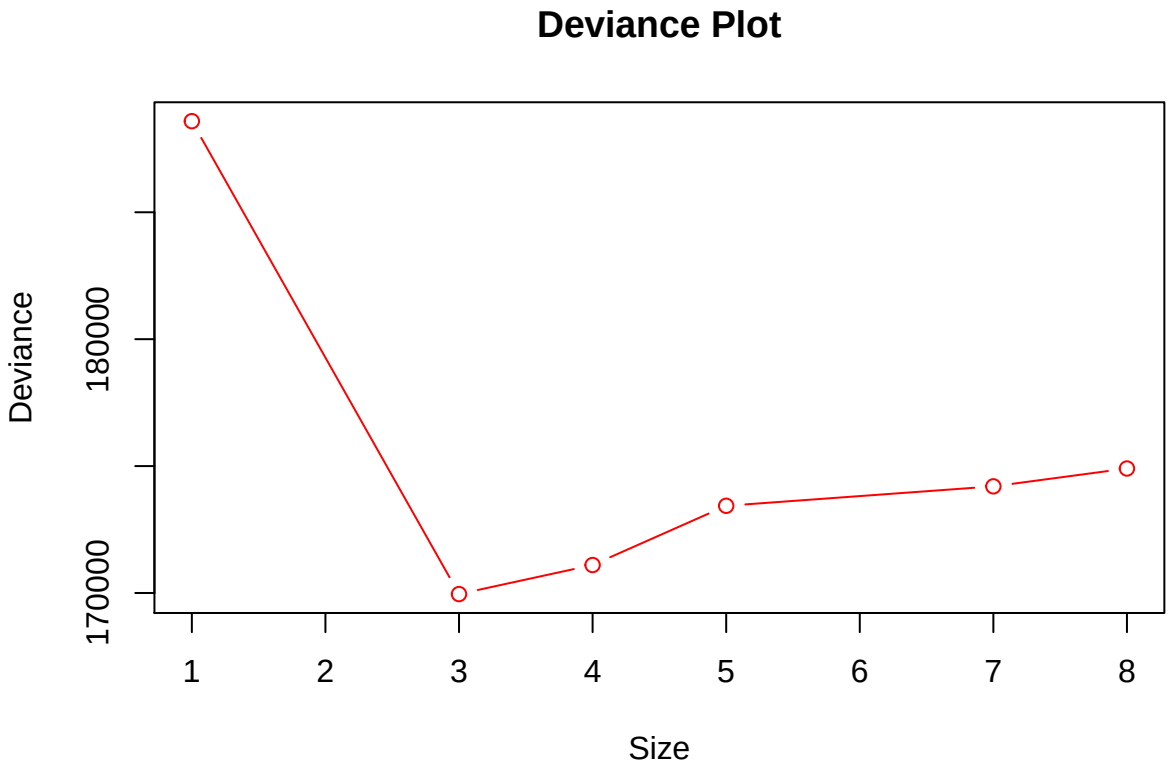
Assignment 3 - Uncertainty estimation

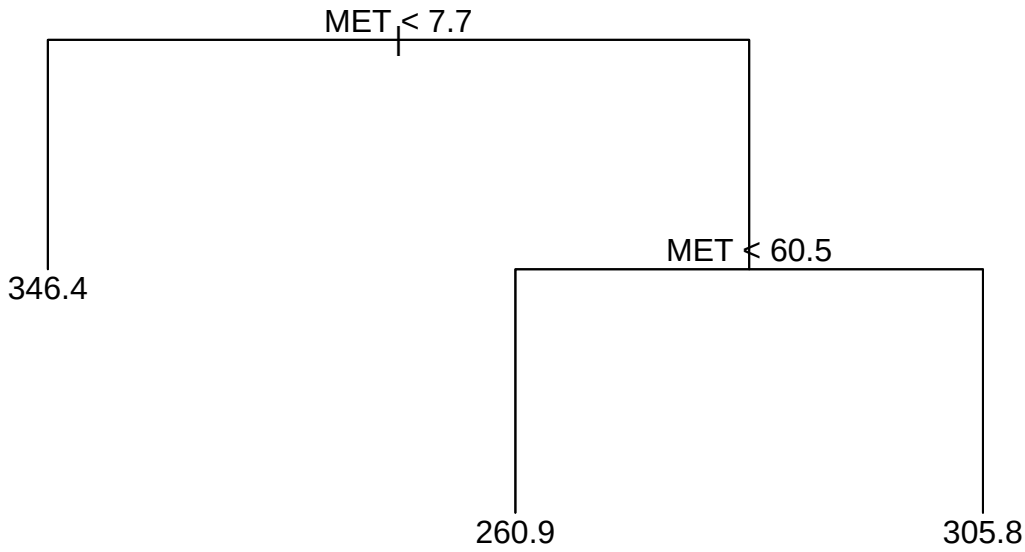
3.1 Plot EX versus MET

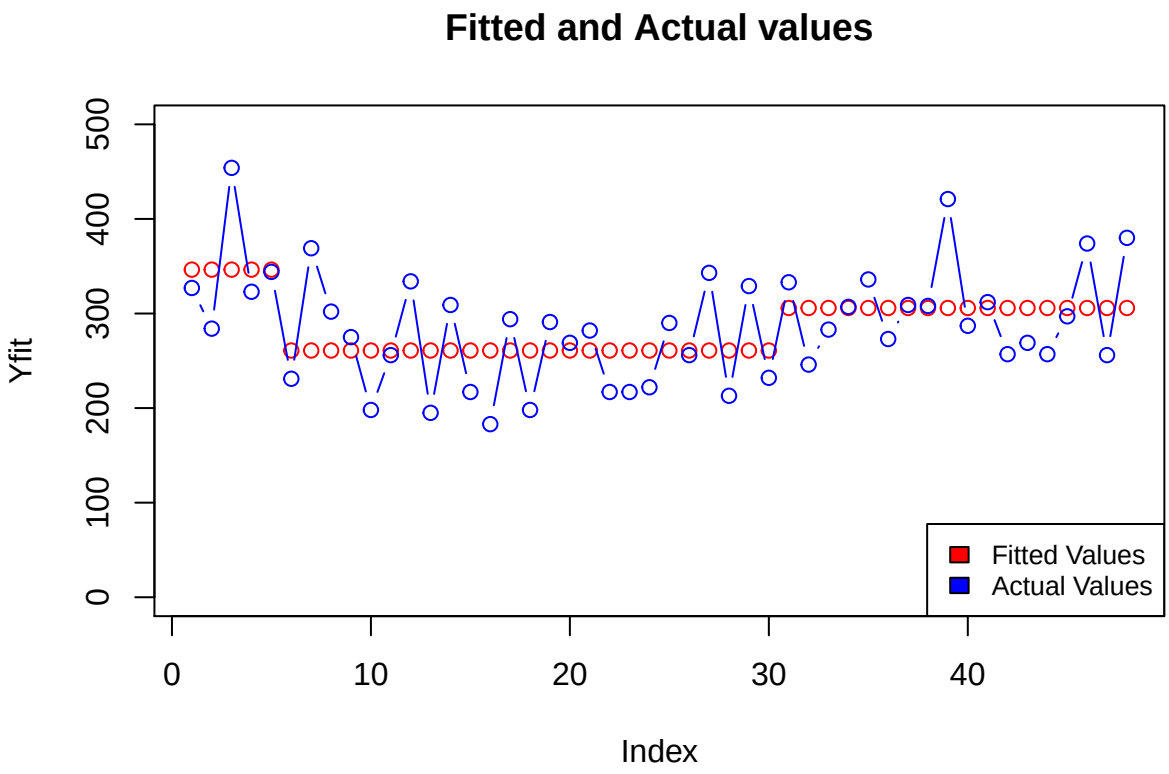


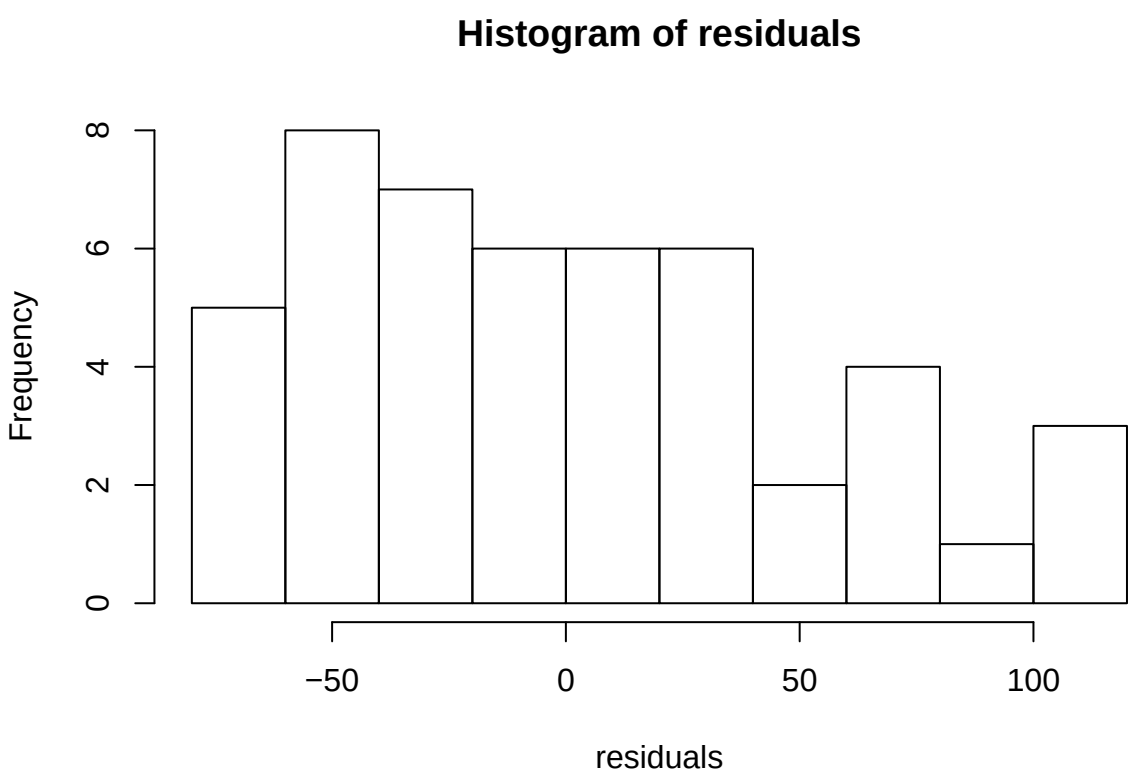
Since, the distribution of the data is not known, bootstrapping technique can be used to model the classifier. Parametric bootstrap may be preferred as the sample size is relatively not large. Given there is only one predictor, it can be simply modelled using a regression tree.

3.2 Fit a regression tree model





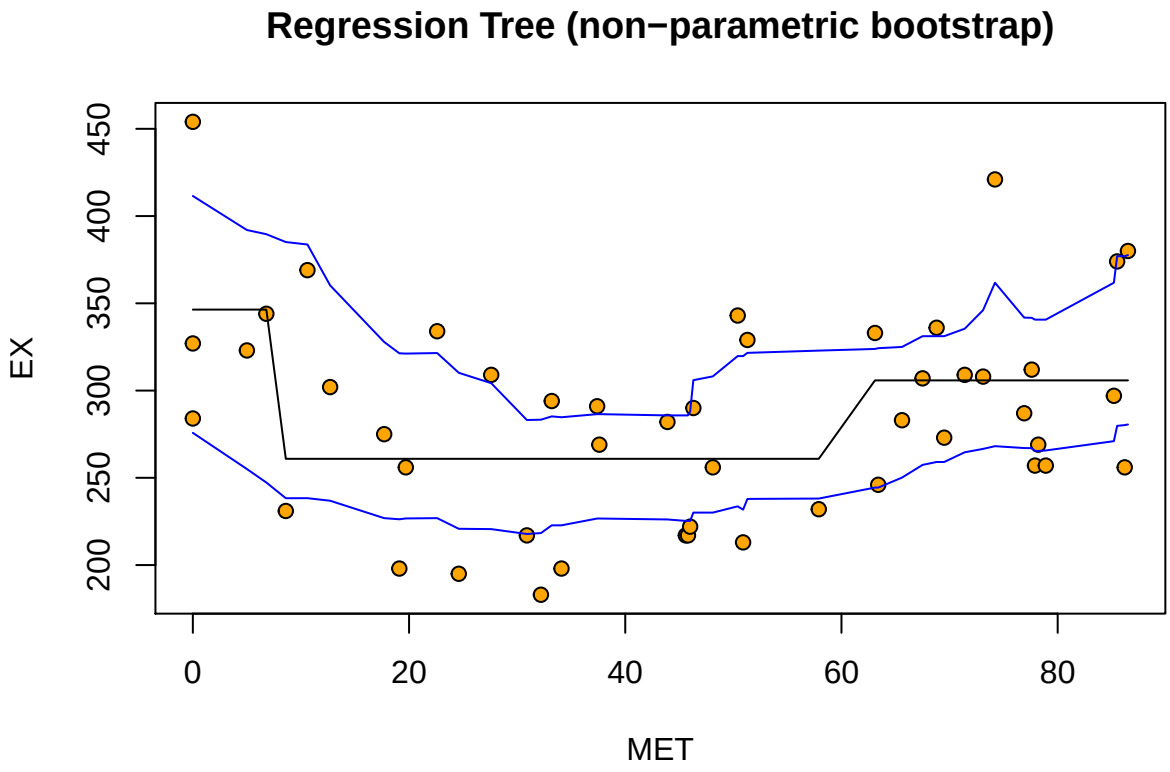

Fitted and Actual values
Yfit
0
100
200
300
400
500
0
10
20
30
40
Index
Fitted Values
Actual Values



Using cross validation, we see from the deviance plot that the regression tree model with 3 terminal nodes and 2 levels (MET < 7.7 and MET < 60.5), has the least deviance. The fitted values are categorized into three estimates based on the optimal regression tree model. Whereas the actual values vary closely around the estimated values.

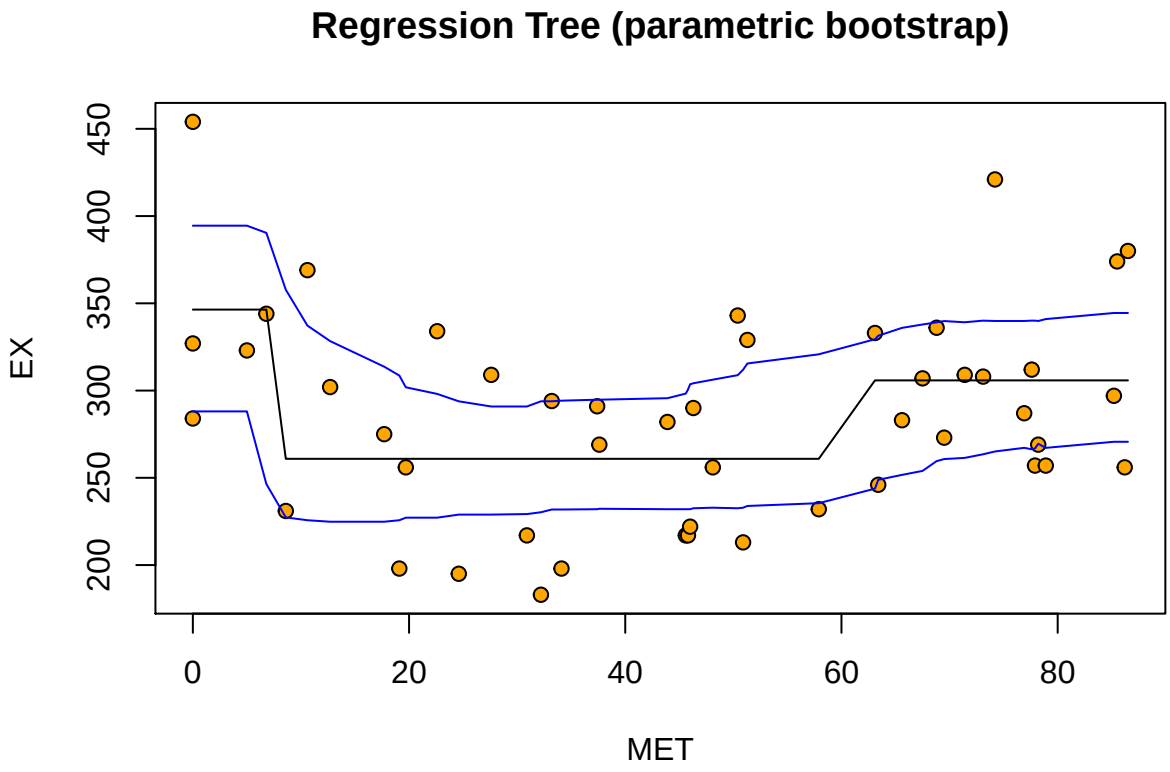
The residuals do not follow any specific distribution. However, it resembles a normal distribution to some extent with a right skewedness. Hence, parametric bootstrap may be used with the estimated mean and variance to model the classifier.

3.3 Plot 95% confidence bands for the regression tree using non-parametric bootstrap



The confidence band is bumpy as the in this case we do not know the distribution of the predicted variableand hence are doing a non paramtric boot-strap. It can also be observed that the confidence interval is approximately 200 units wide which is very large.

3.4 Plot 95% onfidence bands for the regression tree using parametric bootstrap



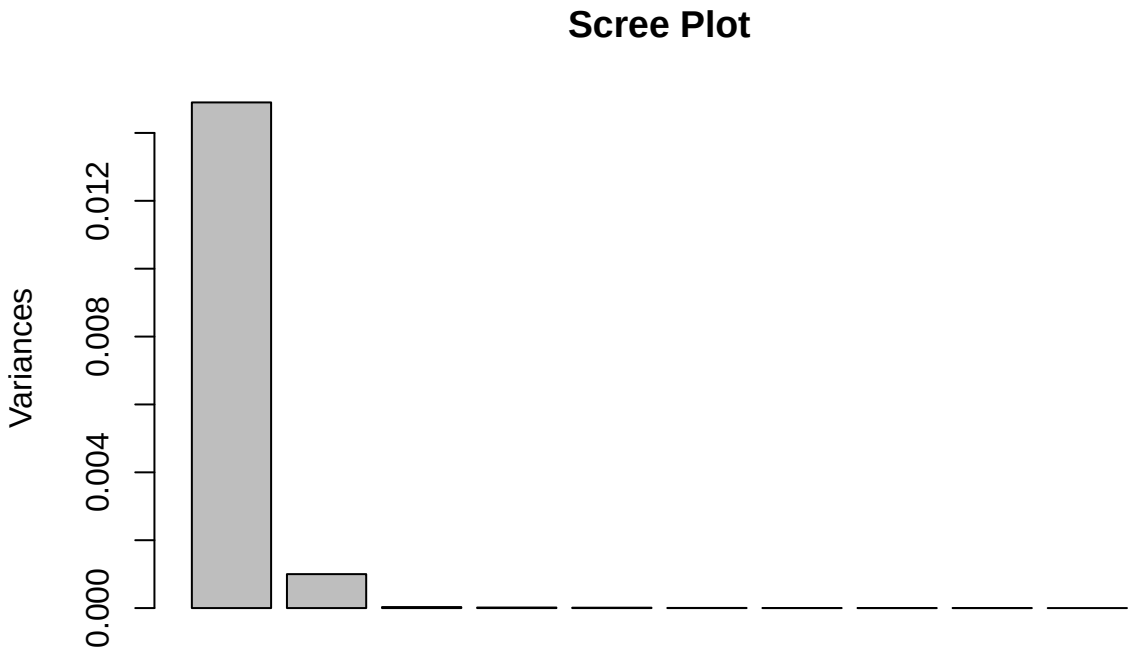
The confidence interval in this case more smooth, making the parametric bootstrap more reliable than the non parametric bootstrap. It looks as if only 5% of the data is outside (i.e 95% is inside) the confidence interval.

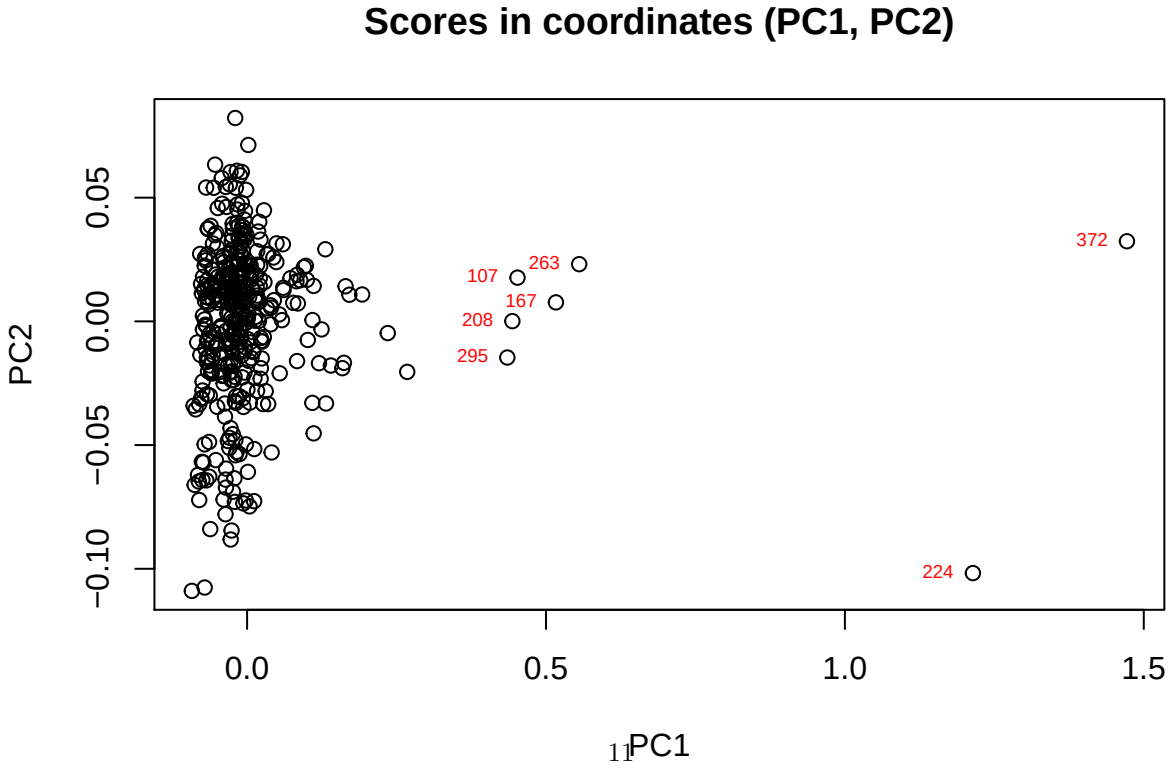
3.5 Comments of boot strapping method

The distribution for the output variable is not known hence it is better to go with a non-parametric bootstrap than the parametric one though the confidence interval is much wider in the non-parametric case.

Assignment 4 - Principal Components

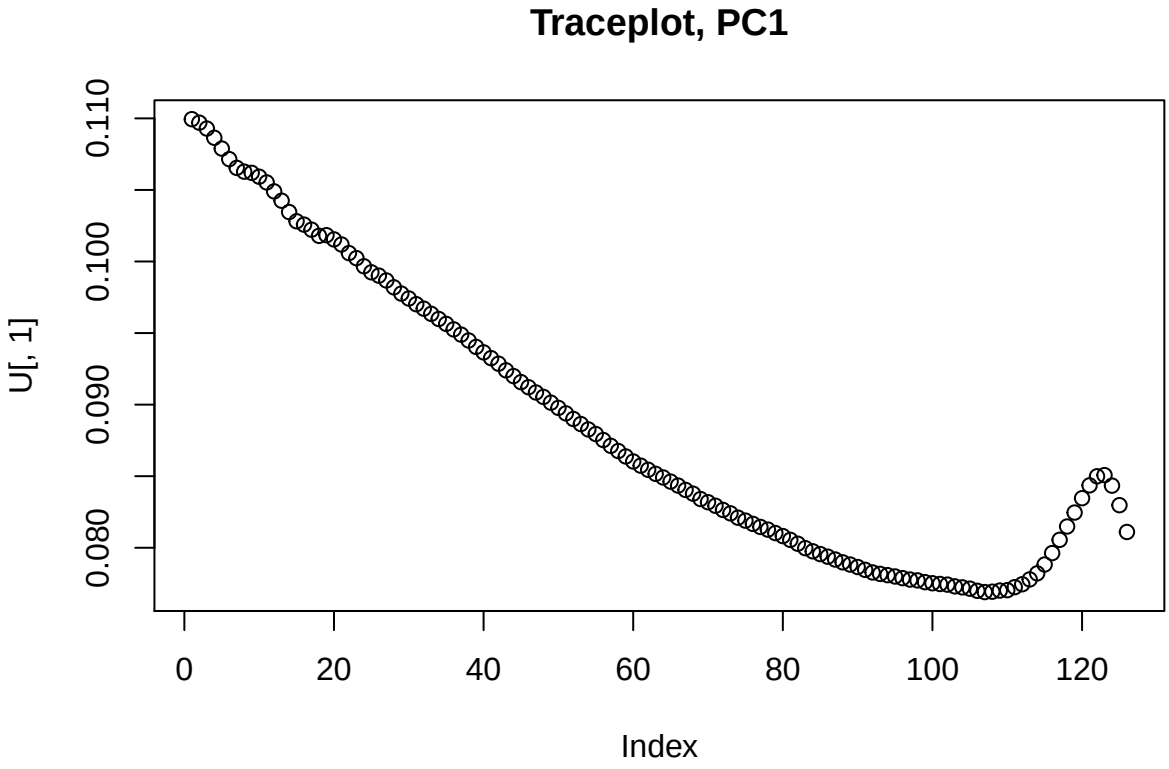
4.1 Standard PCA

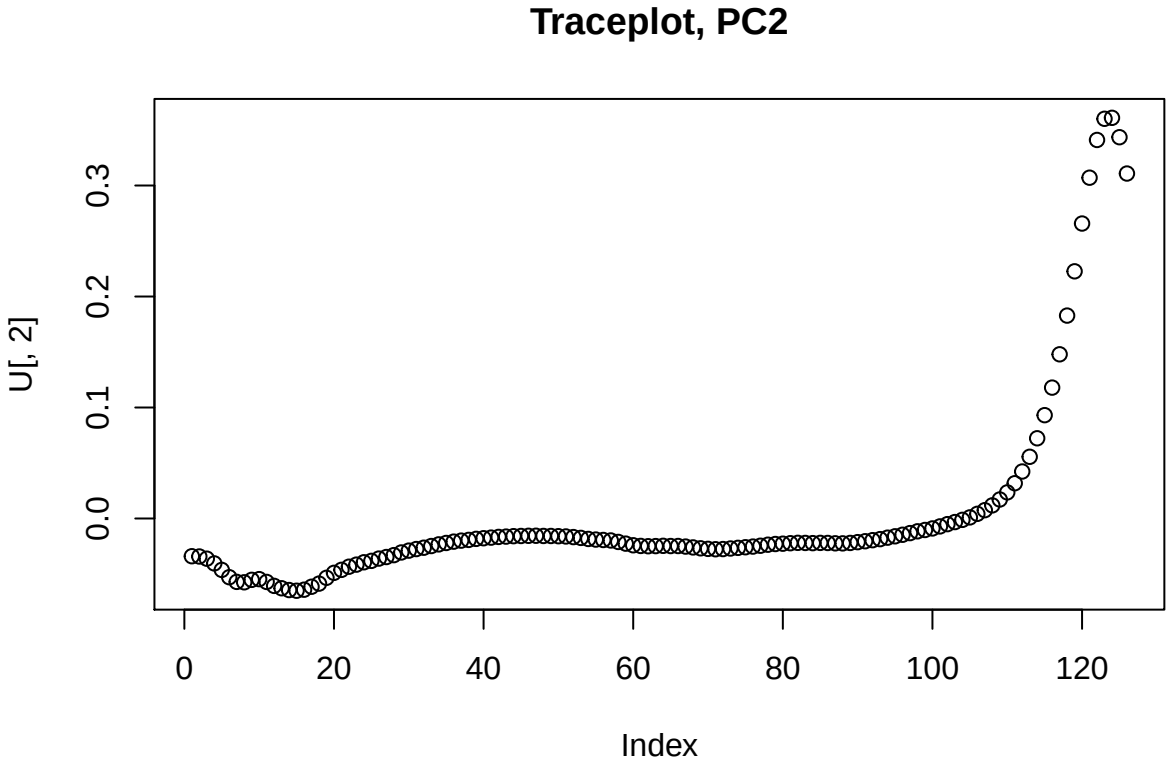




As seen from the screeplot, the first 2 principal components together explain about 99.6 % of the total variance in the data. Since the variance of the projected data along the remaining principal components is very less, only PC1 and PC2 need to be extracted and the rest can be ignored. According to the score plot, we see some of the diesel fuels (labelled in red) that are on the right extreme of the PC1 co-ordinate axis. These unusual values (outliers) cause the high variance along PC1.

4.2 Trace plots of the principal component loadings

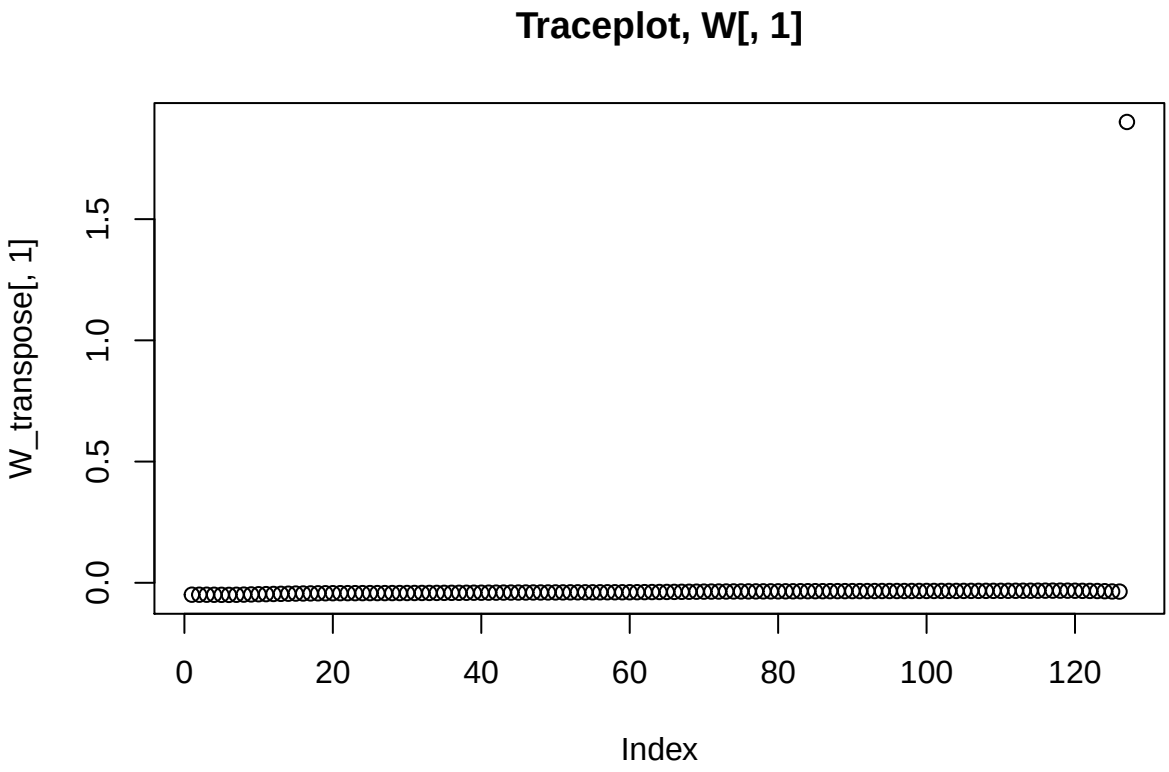


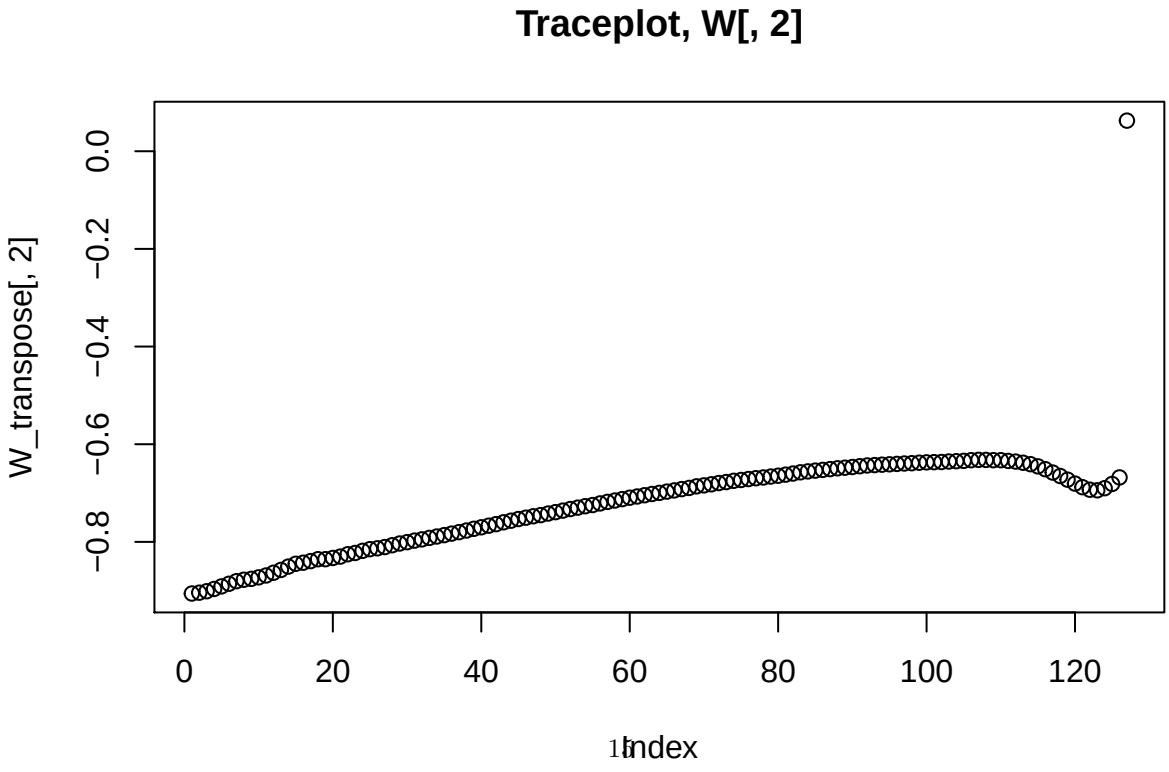


PC1 has the highest variation of about 93.332 and the PC2 has the second highest variation of about 6.263.

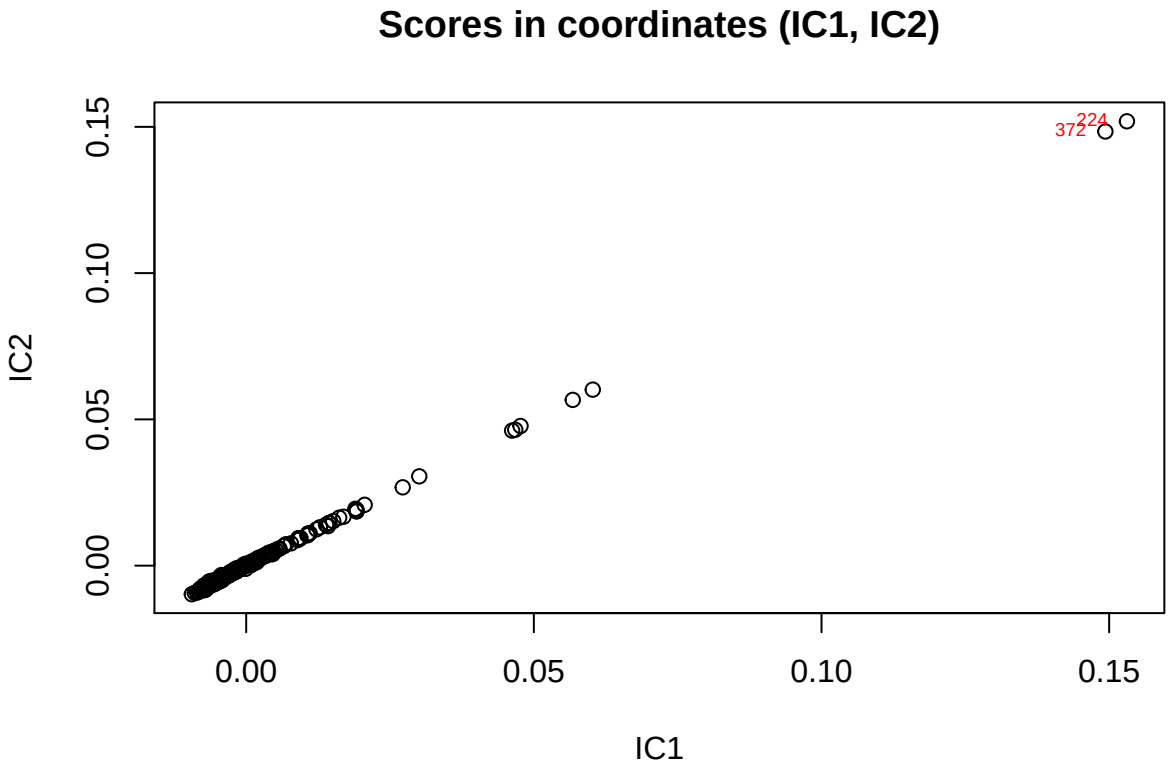
4.3 Independent Component Analysis

4.3 (a) Trace plots of columns of W'





4.3 (b) Scores of the first two latent features



Appendix

```
################################################################################
# Assignment 2 - Analysis of Credit Scoring
################################################################################

# 2.1 Import creditscore data
creditscore <- xlsx::read.xlsx(
  file      = "C:/Users/namit/Downloads/Machine Learning/Lab2 Block1/creditscoring.xls",
  sheetName = "credit",
  header    = TRUE
)

# No of observations in the dataset
n = dim(creditscore)[1]

# Divide dataset into training, validation and test data
RNGversion('3.5.1')
set.seed(12345)
id1 = sample(1:n, floor(n / 2))
train  = creditscore[id1, ]
```

```r
id_rem = setdiff(1:n, id1)

set.seed(12345)
id2   = sample(id_rem, floor(n / 4))
valid = creditscore[id2, ]
id3   = setdiff(id_rem, id2)
test  = creditscore[id3, ]

#-------------------------------------------------------------------------------
# 2.2 Fit a decision tree to training data using different measures of impurity

# 2.2 (a) Deviance
deviance_tree <- tree(formula = good_bad ~ .,
                      data    = train,
                      split   = c("deviance"))

# Predicted values and misclassification rates for train and test datasets
pred_train1    <- predict(object = deviance_tree, newdata = train, type = "class")
cfmat_train1   <- table(actual = train$good_bad, predicted = pred_train1)
mc_rate_train1 <- 1 - (sum(diag(cfmat_train1)) / sum(cfmat_train1))

pred_test1    <- predict(object = deviance_tree, newdata = test, type = "class")
cfmat_test1   <- table(actual = test$good_bad, predicted = pred_test1)
mc_rate_test1 <- 1 - (sum(diag(cfmat_test1)) / sum(cfmat_test1))

# 2.2 (b) Gini index
gini_tree <- tree(formula = good_bad ~ .,
                  data    = train,
                  split   = c("gini"))

# Predicted values and misclassification rates for train and test datasets
pred_train2    <- predict(object = gini_tree, newdata = train, type = "class")
cfmat_train2   <- table(actual = train$good_bad, predicted = pred_train2)
mc_rate_train2 <- 1 - (sum(diag(cfmat_train2)) / sum(cfmat_train2))

pred_test2     <- predict(object = gini_tree, newdata = test, type = "class")
cfmat_test2    <- table(actual = test$good_bad, predicted = pred_test2)
mc_rate_test2  <- 1 - (sum(diag(cfmat_test2)) / sum(cfmat_test2))

#-------------------------------------------------------------------------------
# 2.3 Selecting optimal tree by train and validation
fit <- tree(formula = good_bad ~ ., data = train, split = c("deviance"))

trainScore <- rep(0,9)
testScore  <- rep(0,9)
for(i in 2:9) {
  prunedTree    <- prune.tree(fit, best = i)
  pred          <- predict(prunedTree, newdata = valid, type = "tree")
  trainScore[i] <- deviance(prunedTree)
  testScore[i]  <- deviance(pred)
}

# Plot deviance against tree sizes for train and validation data
```

```r
plot(x    = 2:9,
     y    = trainScore[2:9],
     type = "b",
     col  = "red",
     ylim = c(250,600),
     main = "Deviance Plot",
     xlab = "Number of Leaves",
     ylab = "Deviance")
points(2:9, testScore[2:9], type = "b", col = "blue")
legend("topright",
       legend = c("Training Data", "Validation Data"),
       fill   = c("red", "blue"),
       cex    = 0.8)

# Choose optimal tree depth by pruning
finalTree   <- prune.tree(fit, best = 4)
Yfit        <- predict(object = finalTree, newdata = test, type = "class")
cfmat_test  <- table(actual = test$good_bad, predicted = Yfit)
mc_test     <- 1- (sum(diag(cfmat_test)) / sum(cfmat_test))

tree_summary <- summary(finalTree)  # Summary of the optimal tree

cat("Variables used in tree construction : ",
    paste(tree_summary$used, collapse = ", "),
    "\nOptimal tree depth : ",
    length(tree_summary$used))

# Plot optimal tree
plot(finalTree, main = "Optimial Tree Structure")
text(finalTree, pretty = 0)

#-------------------------------------------------------------------------------
# 2.4 Classification using Naive Bayes
naivebayes <- naiveBayes(good_bad ~ ., data = train)

# Predicted values and misclassification rates for train and test datasets
pred_train    <- predict(object = naivebayes, newdata = train, type = "class")
cfmat_train   <- table(actual = train$good_bad, predicted = pred_train)
mc_rate_train <- 1 - (sum(diag(cfmat_train)) / sum(cfmat_train))

pred_test    <- predict(object = naivebayes, newdata = test, type = "class")
cfmat_test   <- table(actual = test$good_bad, predicted = pred_test)
mc_rate_test <- 1 - (sum(diag(cfmat_test)) / sum(cfmat_test))

#-----------------------------------------------------------------------
# 2.5 Compare Optimal Tree and Naivebayes models and plot ROC

Yfit_optimaltree <- predict(object = finalTree, newdata = test)
Yfit_naivebayes  <- predict(object = naivebayes, newdata = test, type = "raw")

Yfit_optimaltree <- as.data.frame(Yfit_optimaltree)
Yfit_naivebayes  <- as.data.frame(Yfit_naivebayes)
```

```r
TPR <- data.frame(ot = numeric(19),           # TPR for optimal tree
                  nb = numeric(19))           # TPR for naive bayes
FPR <- data.frame(ot = numeric(19),           # FPR for optimal tree
                  nb = numeric(19))           # FPR for naive bayes

i <- 1
for (pi in seq(0.05, 0.95, 0.05)) {
  # Apply Classification principle on predictions from both models
  Yfit_ot  <- ifelse(Yfit_optimaltree$good > pi, 1, 0)
  Yfit_nb  <- ifelse(Yfit_naivebayes$good > pi, 1, 0)

  # Confusion matrices for the two models
  cfmat_ot <- table(test$good_bad, Yfit_ot)
  cfmat_nb <- table(test$good_bad, Yfit_nb)

  tryCatch({
      # True positive rates for the two models
      TPR$ot[i] <- cfmat_ot["good", "1"] / sum(cfmat_ot["good", ])
      TPR$nb[i] <- cfmat_nb["good", "1"] / sum(cfmat_nb["good", ])

      # False positive rates for the two models
      FPR$ot[i] <- cfmat_ot["bad", "1"] / sum(cfmat_ot["bad", ])
      FPR$nb[i] <- cfmat_nb["bad", "1"] / sum(cfmat_nb["bad", ])

      i <- i + 1
    },
    error = function(e) {}
  )
}

# Plot the ROC curves for both optimal tree and naive bayes models
plot(x    = FPR$ot,
     y    = TPR$ot,
     type = "o",
     col  = "red",
     main = "ROC Curves",
     xlab = "FPR",
     ylab = "TPR")
points(FPR$nb, TPR$nb, type = "o", col = "blue")
legend("topleft",
       legend = c("Regression Tree", "Naive Bayes"),
       fill   = c("red", "blue"),
       cex    = 0.8)

#-------------------------------------------------------------------------
# 2.6 Naive Bayes classification with a loss matrix
L <- data.frame(pred.bad  = c(0, 1),
                pred.good = c(10, 0),
                row.names = c("obs.bad", "obs.good"))

naivebayes <- naiveBayes(good_bad ~ ., data = train)
Yfit       <- predict(object = naivebayes, newdata = test, type = "raw")
Yfit_loss  <- data.frame(bad  = as.data.frame(Yfit)$bad * 10,
```

```
                          good = as.data.frame(Yfit)$good * 1)
Yfit_val   <- ifelse(Yfit_loss$good > Yfit_loss$bad, "good", "bad")
cfmat      <- table(actual = test$good_bad, predicted = Yfit_val)
mc_rate    <- 1 - (sum(diag(cfmat)) / sum(cfmat))

# False positive rate comparison
FPR        <- c(cfmat["bad", "good"] / sum(cfmat["bad", ]),
                cfmat_test["bad", "good"] / sum(cfmat_test["bad", ]))

##############################################################################
# Assignment 3 - Uncertainty estimation
##############################################################################

# Import state data
state <- read.csv2(
  file      = "C:/Users/namit/Downloads/Machine Learning/Lab2 Block1/State.csv",
  header    = TRUE
)

# No of observations in the dataset
n <- dim(state)[1]

#-----------------------------------------------------------------------------
# 3.1 Plot EX versus MET
state_ord <- state[order(state$MET), ]
plot(x    = state_ord$MET,
     y    = state_ord$EX,
     col  = "blue",
     main = "EX vs MET",
     xlab = "% population in standard metropolitan areas",
     ylab = "Per capita state and local public expenditures ($)")

#-----------------------------------------------------------------------------
# 3.2 Fit a regression tree model
regtree <- tree(formula = EX ~ MET,
                data    = state_ord,
                control = tree.control(nobs = n, minsize = 8))

cvtree <- cv.tree(regtree)
plot(x = cvtree$size,
     y = cvtree$dev,
     type = "b",
     col  = "red",
     main = "Deviance Plot",
     xlab = "Size",
     ylab = "Deviance")

# Optimized regression tree
finalTree <- prune.tree(regtree, best = 3)
plot(finalTree)
text(finalTree, pretty = 0)

# Predictions and residuals
```

```r
Yfit      <- predict(object = finalTree, newdata = state_ord)
residuals <- residuals(finalTree)

# Plot fitted values and actual values
plot(Yfit, col = "red", ylim = c(0,500), main = "Fitted and Actual values")
points(state_ord$EX, type = "b", col = "blue")
legend("bottomright",
       legend = c("Fitted Values", "Actual Values"),
       fill   = c("red", "blue"),
       cex    = 0.8)

# Plot Residuals
hist(residuals)
#-------------------------------------------------------------------------------
# 3.3 Plot 95% onfidence bands for the regression tree using non-parametric bootstrap

# computing bootstrap samples
f_np <- function(data, ind){
  # Extract bootstrap sample
  bs_sample <- data[ind, ]

  # Fit regression tree
  regtree <- prune.tree(tree(EX ~ MET, bs_sample, minsize = 8), best = 3)
  # Predict EX for all MET from the original data
  exPredict = predict(regtree, newdata = state_ord)
  return(exPredict)
}

# Make bootstrap
np_bootstrap <- boot(state_ord, f_np, R = 1000)

# Compute confidence bands
cf_band   <- envelope(np_bootstrap)
regtree   <- prune.tree(tree(EX ~ MET, state_ord, minsize = 8), best = 3)
exPredict <- predict(regtree)

# Plot fitted line
plot(x    = state_ord$MET,
     y    = state_ord$EX,
     main = "Regression Tree (non-parametric bootstrap)",
     xlab = "MET",
     ylab = "EX",
     pch  = 21,
     bg   = "orange")
points(state_ord$MET, exPredict, type = "l")

# Plot cofidence bands
points(state_ord$MET, cf_band$point[2,], type = "l", col = "blue")
points(state_ord$MET, cf_band$point[1,], type = "l", col = "blue")

#-------------------------------------------------------------------------------
# 3.4 Plot 95% onfidence bands for the regression tree using parametric bootstrap
MLE <- prune.tree(tree(EX ~ MET, state_ord, minsize = 8), best = 3)
```

```r
RNG <- function(data, mle) {
  datanew <- data.frame(EX = data$EX, MET = data$MET)
  n       <- length(data$EX)
  # Generate new Expenditure
  datanew$EX <- rnorm(n, predict(MLE, newdata = datanew), sd(residuals(MLE)))
  return(datanew)
}

fp <- function(data){
  # Fit regression tree
  regtree <- prune.tree(tree(EX ~ MET, data, minsize = 8), best = 3)
  # Predict values for all MET values from the original data
  exPredict <- predict(regtree, newdata = state_ord)
  return(exPredict)
}

# Make bootstrap
p_bootstrap = boot(state_ord, statistic = fp, R = 1000, mle = MLE, ran.gen = RNG, sim = "parametric")

# Compute confidence bands
cf_band   <- envelope(p_bootstrap)
regtree   <- prune.tree(tree(EX ~ MET, state_ord, minsize = 8), best = 3)
exPredict <- predict(regtree)

# Plot fitted line
plot(x    = state_ord$MET,
     y    = state_ord$EX,
     main = "Regression Tree (parametric bootstrap)",
     xlab = "MET",
     ylab = "EX",
     pch  = 21,
     bg   = "orange")
points(state_ord$MET, exPredict, type = "l")

# Plot cofidence bands
points(state_ord$MET, cf_band$point[2,], type = "l", col = "blue")
points(state_ord$MET, cf_band$point[1,], type = "l", col = "blue")

################################################################################
# Assignment 4 - Principal Components
################################################################################

# Import near-infrared spectra data
NIRSpectra <- read.csv2(
  file      = "C:/Users/namit/Downloads/Machine Learning/Lab2 Block1/NIRSpectra.csv",
  header    = TRUE
)

# No of observations in the dataset
n <- dim(NIRSpectra)[1]

#-------------------------------------------------------------------------------
# 4.1 Standard PCA
```

```r
PC        <- prcomp(NIRSpectra[, -127])                    # PCA
lambda    <- PC$sdev ^ 2                                   # Eigenvalues
variation <- sprintf("%2.3f", lambda / sum(lambda) * 100) # Proportion of variation
screeplot(PC, main = "Scree Plot")

# Unusual Diesel Fuels
unusual <- ifelse(PC$x[, 1] > 0.4, 1:395, as.numeric(NA))

# Data in (PC1, PC2) - Scores (z)
plot(x      = PC$x[, 1],
     y      = PC$x[, 2],
     main   = "Scores in coordinates (PC1, PC2)",
     xlab   = "PC1",
     ylab   = "PC2")
text(x      = PC$x[, 1],
     y      = PC$x[, 2],
     labels = unusual[unusual != "NA"],
     pos    = 2,
     col    = "red",
     cex    = 0.6)

#-------------------------------------------------------------------------------
# 4.2 Trace plots of the principal component loadings
U <- PC$rotation
plot(U[, 1], main  ="Traceplot, PC1")
plot(U[, 2], main = "Traceplot, PC2")

#-------------------------------------------------------------------------------
# 4.3 Independent Component Analysis
set.seed (12345)
IC          <- fastICA(NIRSpectra, 2)                      # ICA
W_transpose <- IC$K %*% IC$W                               # Estimated un-mixing matrix

# 4.3 (a) Trace plots of columns of W'
plot(W_transpose[, 1], main = "Traceplot, W[, 1]")
plot(W_transpose[, 2], main = "Traceplot, W[, 2]")

# 4.3 (b) Scores of the first two latent features

# Unusual Diesel Fuels
unusual <- ifelse(IC$X[, 1] > 0.10, 1:395, as.numeric(NA))

plot(x      = IC$X[, 1],
     y      = IC$X[, 2],
     main   = "Scores in coordinates (IC1, IC2)",
     xlab   = "IC1",
     ylab   = "IC2")
text(x      = IC$X[, 1],
     y      = IC$X[, 2],
     labels = unusual[unusual != "NA"],
     pos    = 2,
     col    = "red",
     cex    = 0.6)
```